|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №1**

по дисциплине «Интерфейсы прикладного программирования»

**Тема практической работы:**

**Знакомство с публичными WEB API**

**Студент группы** ИКБО-01-21 Маров Герман Андреевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** ассистент Зарипов Е.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

**ОГЛАВЛЕНИЕ**

[Цель работы 3](#_Toc114065479)

[Ход работы 3](#_Toc114065480)

[Тестирование 6](#_Toc114065481)

[Вывод 6](#_Toc114065482)

# Цель работы

Целью данной работы является первичное знакомство студентов с интерфейсами прикладного программирования и их применением на практике при разработке программных систем.

# Ход работы

В качестве сервиса, предоставляющего API используется HERE Platform – аналог Google Maps, предоставляющий геолокационную информацию.

Листинг 1 – Класс Location.java

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@ToString

@Builder

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

public class Location {

private Double latitude = 0d;

private Double longitude = 0d;

private String countryCode = "";

private String countryName = "";

private String county = "";

private String city = "";

private String district = "";

private String street = "";

private String houseNumber = "";

private String description = "";

}

Листинг 2 – Класс Finder.java

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.IOException;

import java.net.URI;

import java.net.URLEncoder;

import java.net.http.HttpClient;

import java.net.http.HttpRequest;

import java.net.http.HttpResponse;

import java.nio.charset.StandardCharsets;

import java.time.Duration;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Finder {

private static final String GEOCODING\_RESOURCE = "https://geocode.search.hereapi.com/v1/geocode";

public static final String REVERSE\_GEOCODING\_RESOURCE = "https://revgeocode.search.hereapi.com/v1/revgeocode";

private static final String API\_KEY = "wLdioPNGe6VILOPfX6Je\_sL9REZqesRYJy7Nx5H0Mq4";

public static Location getLocationFromCoordinates(Double latitude, Double longitude) {

try {

String geocode = getGeocodeResponseFromCoordinates(latitude, longitude);

ObjectMapper mapper = new ObjectMapper();

JsonNode responseJsonNode = mapper.readTree(geocode);

JsonNode items = responseJsonNode.get("items");

return getLocationFromJsonNode(items.get(0));

} catch (IOException | InterruptedException e) {

throw new RuntimeException(e);

}

}

public static List<Location> getLocationFromQuery(String query) {

try {

String geocode = getGeocodeResponseFromQuery(query);

ObjectMapper mapper = new ObjectMapper();

JsonNode responseJsonNode = mapper.readTree(geocode);

JsonNode items = responseJsonNode.get("items");

List<Location> resultList = new ArrayList<>();

items.forEach(item -> resultList.add(getLocationFromJsonNode(item)));

return resultList;

} catch (IOException | InterruptedException e) {

throw new RuntimeException(e);

}

}

private static String getGeocodeResponseFromQuery(String query) throws IOException, InterruptedException {

HttpClient httpClient = HttpClient.newHttpClient();

String encodedQuery = URLEncoder.encode(query, StandardCharsets.UTF\_8);

String requestUri = GEOCODING\_RESOURCE

+ "?apiKey=" + API\_KEY

+ "&q=" + encodedQuery

+ "&lang=ru-RUS";

HttpRequest geocodingRequest = HttpRequest.newBuilder()

.GET()

.uri(URI.create(requestUri))

.timeout(Duration.ofMillis(5000))

.build();

HttpResponse<String> geocodingResponse = httpClient.send(geocodingRequest, HttpResponse.BodyHandlers.ofString());

return geocodingResponse.body();

}

private static String getGeocodeResponseFromCoordinates(Double latitude, Double longitude) throws IOException, InterruptedException {

HttpClient httpClient = HttpClient.newHttpClient();

String requestUri = REVERSE\_GEOCODING\_RESOURCE

+ "?apiKey=" + API\_KEY

+ "&at=" + latitude + "," + longitude

+ "&lang=ru-RUS";

HttpRequest geocodingRequest = HttpRequest.newBuilder()

.GET()

.uri(URI.create(requestUri))

.timeout(Duration.ofMillis(5000))

.build();

HttpResponse<String> geocodingResponse = httpClient.send(geocodingRequest, HttpResponse.BodyHandlers.ofString());

return geocodingResponse.body();

}

private static Location getLocationFromJsonNode(JsonNode node) {

JsonNode address = node.get("address");

JsonNode position = node.get("position");

String label = Objects.isNull(address.get("label")) ? "" : address.get("label").asText();

String countryCode = Objects.isNull(address.get("countryCode")) ? "" : address.get("countryCode").asText();

String countryName = Objects.isNull(address.get("countryName")) ? "" : address.get("countryName").asText();

String county = Objects.isNull(address.get("county")) ? "" : address.get("county").asText();

String city = Objects.isNull(address.get("city")) ? "" : address.get("city").asText();

String district = Objects.isNull(address.get("district")) ? "" : address.get("district").asText();

String street = Objects.isNull(address.get("street")) ? "" : address.get("street").asText();

String houseNumber = Objects.isNull(address.get("houseNumber")) ? "" : address.get("houseNumber").asText();

return Location.builder()

.latitude(position.get("lat").asDouble())

.longitude(position.get("lng").asDouble())

.countryCode(countryCode)

.countryName(countryName)

.county(county)

.city(city)

.district(district)

.street(street)

.houseNumber(houseNumber)

.description(label)

.build();

}

}

Листинг 3 – Класс App.java

public class App {

public static void main(String[] args) {

System.out.println(Finder.getLocationFromCoordinates(55.67, 37.48).toString() + "\n");

Finder.getLocationFromQuery("Москва, МИРЭА - Российский технологический университет")

.forEach(item -> System.out.println(item.toString()));

}

}

# Тестирование

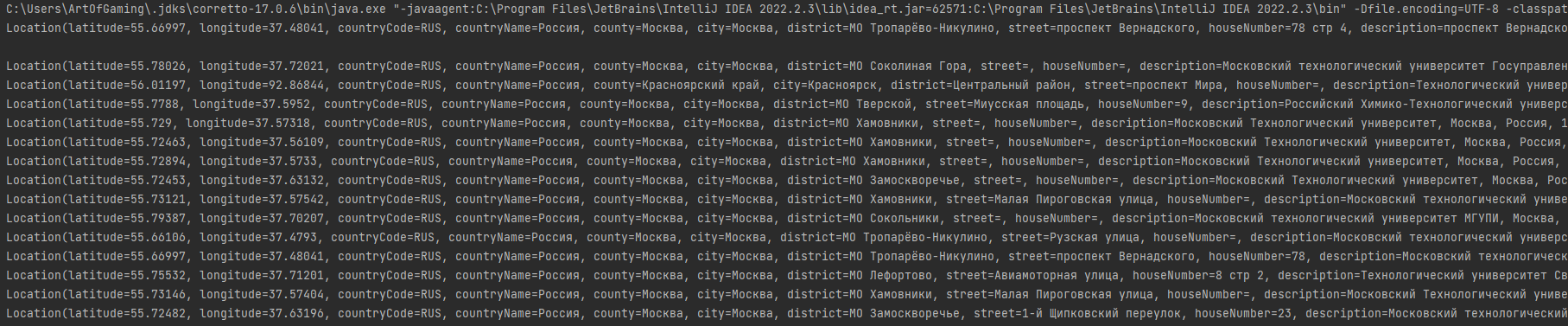


Рисунок 1 – Результат, возвращенный при запросе к HERE API

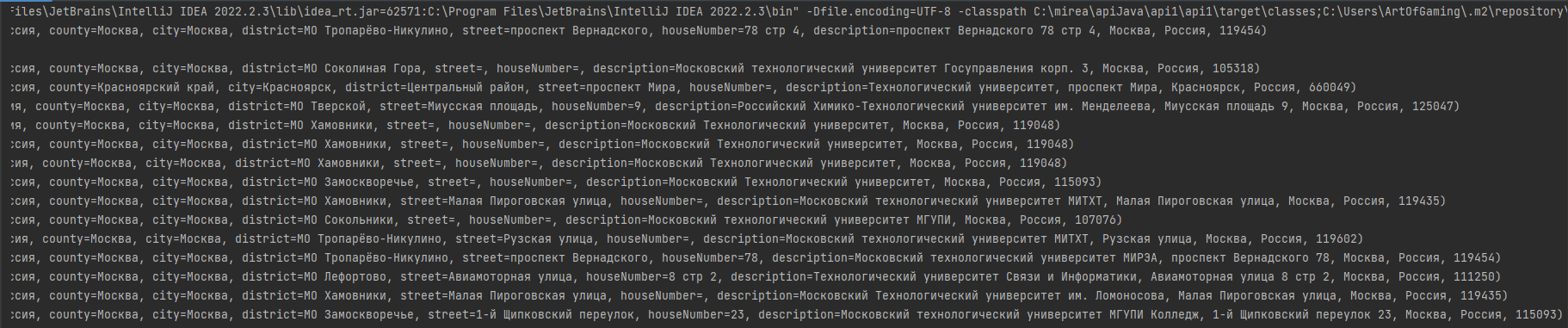


Рисунок 2 – Результат, возвращенный при запросе к HERE API

# Вывод

В результате выполнения данной практической работы было проведено первичное знакомство с интерфейсами прикладного программирования и их применением при разработке программных систем на примере публичного API платформы HERE.